COMPASS CTF Tutorial 5: Encoding and Cryptography

COMPASS CTF 教程【5】：编码与密码学

30016794 Zhao, Li (Research Assistant)

COMPuter And System Security Lab, Computer Science and Technology Department, College of Engineering (CE), SUSTech University.
南方科技大学 工学院 计算机科学与技术系 计算机与系统安全实验室

2023 年 8 月 14 日

密码学概述

密码学是我们能够使用银行应用，在网络上传输敏感信息，以及通常保护我们隐私的原因。然而，CTFs 的很大一部分是破坏被错误实现的广泛使用的加密方案。这些数学问题看起来可能令人生畏，但是，更多时候，对基本原理的简单理解会让你找到漏洞，破解代码。

“密码学”这个词技术上来讲意味着编写代码的艺术。当涉及到数字取证时，它是你用来理解数据是如何为你的分析而构建的一种方法。

密码学的用途

日常软件中的用途

1. 确保网络流量的安全（密码，通信等）
2. 保护版权的软件代码

恶意用途

1. 隐藏恶意通信
2. 隐藏恶意代码

异或

『异或』又被称为『非等价』，其是等价的否定，是一种逻辑操作，当且仅当其参数不同（一个为真，另一个为假）时才为真。
它之所以被命名为『异或』，是因为当两个操作数都为真时，”或”的含义会产生歧义；异或操作符排除了这种情况。这有时被看作是“一个或另一个，但不是两者都是”，或者“是这个，或者是那个”。这可以被写成“A 或 B，但不是 A 和 B”。
XOR 等价于逻辑不等式（NEQ），因为它只有在输入不同时（一个为真，一个为假）才为真。XOR 的否定是逻辑双条件词，只有当两个输入相同时才为真，这等价于逻辑等式（EQ）。
由于它是关联的，它可能被视为一个 n 元运算符，当且仅当奇数个参数为真时才为真。也就是说，a XOR b XOR ... 可能被视为 XOR(a,b,...)。

真值表

A	B	A xor B
False	False	False
False	True	True
True	False	True
True	True	False

异或运算在计算机科学中的应用

在计算机科学中，异或运算有多种应用：

- 告知两位是否不同
- 可选择地翻转位（决定输入选择是否翻转数据输入）。
- 指示 1 的个数是否为奇数，这等同于奇偶函数返回的奇偶位。

异或运算的其他应用

异或运算也常在如 AES (Rijndael) 或 Serpent 这样的块密码和块密码实现 (CBC, CFB, OFB 或 CTR) 中使用。
异或运算可用于生成硬件随机数生成器的熵池。异或操作保留随机性，意味着一个随机位与一个非随机位进行异或运算将得到一个随机位。
异或运算被用于 RAID 3 – 6 用以创建奇偶校验信息。
异或运算也用于检测有符号二进制算术操作结果的溢出。
XOR linked lists 运用了异或运算的性质来节省表示双向链表数据结构的空间。
在计算机图形学中，基于异或的绘图方法常用于无 alpha 通道或叠加平面的系统上管理如边界框和光标等项目。

代换密码

在密码学中，代换密码是一种加密方法，其中明文的单元被以定义的方式用帮助密钥替换为密文。这些“单元”可能是单个字母（最常见）、字母对、三字母和以上种类的混合等。接收者通过执行逆转代换过程来解密文本，以提取原始消息。[1]
代换密码可以与置换密码进行比较。在置换密码中，明文的单元在一个不同且通常复杂的顺序中被重新排列，但单元本身没有改变。相比之下，换密码中，明文的各单元在密文中以相同的序列保持，但是各单元本身被改变。

代换密码

代换密码有很多不同的类型。如果密码运算在单个字母上，它被称为简单代换密码；一个运算在较大字母组的密码被称为多图形密码。单字母密码在整个消息中使用固定的代换，而多字母密码在消息的不同位置使用多个代换，其中明文的一个单元被映射到密文的几种可能性之一，并且反之亦然。
第一次公开发表如何破解简单代换密码的描述是由 Al-Kindi 在大约 850 年的《关于解密密码消息的手稿》中给出的。他描述的方法现在被称为频率分析。

简单替换

ROT13 是凯撒密码的一种，也是替换密码的一种形式。在 ROT13 中，字母表沿一个方向旋转 13 个步骤。单个字母的替换——简单替换——可以通过以某种顺序写出字母表以表示替换来演示。此操作被称为替换字母表。密码字母表可能被移位或反转（形成凯撒和 Atbash 密码），或者以更复杂的方式打乱，此时它被称为混淆字母表或混乱字母表。

在这个系统中，关键词 "zebras" 会给出以下字母表：

Plaintext alphabet	ABCDEFGHIJKLMNOPQRSTUVWXYZ
Ciphertext alphabet	ZEBRASCDFGHIJKLMNOPQTUVWXY

简单替换

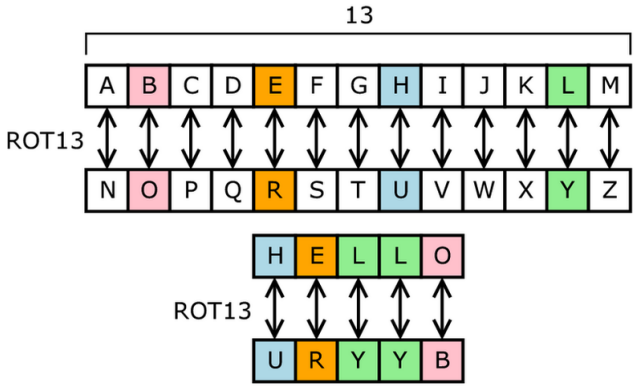

图: 代换密码

简单替换

信息 "flee at once. we are discovered!" 会被加密为 "SIAA ZQ LKBA. VA ZOA RFPBLUAOAR!"。通常情况下，密文会按固定长度的块写出，省略标点和空格，这样做是为了从明文中掩盖单词边界，帮助避免传输错误。这些块被称为" 组"，并且有时候，作为一个额外的检查，会给出 " 组数"（即组的数量）。五字母组经常被使用，这个习惯来源于以前电报传输信息的时候。
如果消息的长度恰好不能被五整除，那么它可能会在最后用 "nulls" 填充。这些可以是解密后明显无意义的任何字符，使得接收者可以轻松地发现并丢弃它们。

简单替换

明文和密文的字母表有时会有所不同。例如，在猪圈密码中，密文由来自网格的一组符号组成。然而，这些特征对一个方案的安全性影响很小 — 至少，任何奇特的符号都可以被转录回 A-Z 字母表并作正常处理。

图: 猪圈密码

例如说，MAT 会被用来表示 120。[2]

Plaintext digits	1234567890
Ciphertext alphabets	MAKEPROFIT

混合代替字母表的生成方式的安全性

尽管使用传统的关键词方法来创建混合替代字母表很简单，但一个严重的缺点是字母表的最后几个字母（其主要是低频字母）往往会留在最后。构造混合字母表的更强方法是完全随机生成替代字母表。
尽管可能的替代字母表的数量非常大（$26! \approx 288.4$，等于大约 88 位），但这种密码并不很强，容易被破译。只要消息长度适中（如下文），密码分析师可以通过分析密文的频率分布来推断出最常见符号的可能含义。这允许形成部分单词，可以尝试着填补，逐步扩展（部分）解决方案（对此的演示可以参见频率分析）。

混合代替字母表的生成方式的安全性

在某些情况下，还可以根据它们的字母模式确定潜在的单词；例如，attract，osseous，和以这两个单词作为词根的词是唯一常见的英文单词，其字母模式为 ABBCADB。许多人将解决此类密码作为消遣，就像在报纸上解密程序的谜语一样。根据英语的唯一性距离，需要 27.6 个密文字母来破译一个混合字母表的简单替代。在实践中，通常需要大约 50 个字母，尽管如果找到了不寻常的模式，可以使用更少的字母来解码一些信息。在其他情况下，可以设计明文使其频率分布接近于平坦，然后密码分析师将需要更长的明文。

Vigenère 密码的介绍

Vigenère 密码是一种用于加密字母文本的方法，其中明文的每个字母都使用不同的凯撒密码进行编码，其增量由另一个文本（即密钥）的相应字母确定。

图: 维吉尼亚

一个例子

例如，如果明文是"attacking tonight"，密钥是"OCULORHINOLARINGOLOGY"，那么明文的第一个字母'a' 将向前移动 14 个位置（因为密钥的第一个字母'O' 是字母表的第 14 个字母），得到'o'；第二个字母't' 前移 2 个位置（密钥的第二个字母'C' 对应 2），得到'v'。不断重复这一过程，我们得到的密文是"ovnlqbpvt eoeqtnh"。如果消息的接收者知道密钥，他们可以通过反转这个过程来恢复明文。

历史和分析

Vigenère 密码是多字母替代法的一个特殊情况[3][4]，最早由 Giovan Battista Bellaso 在 1553 年描述，这种密码易于理解和实现[5]，但直到三个世纪后的 1863 年，人们才找到破解它的方法。这使它获得了“不可破译密码”的描述。许多人尝试实现基本上是 Vigenère 密码的加密方案。1863 年，Friedrich Kasiski 首次公开发布了解密 Vigenère 密码的通用方法。

在 19 世纪，这种方案被误归因于 Blaise de Vigenère（1523-1596），因此得名。[6]

凯撒密码的基本原理

在凯撒密码中，字母表中的每一个字母会向前移动一定数量的位置。例如，偏移量为 3 的凯撒密码中，a 会变成 d，b 会变成 e，y 会变成 b 等。维吉尼亚密码实际上是多个凯撒密码的连续使用，其偏移量也各不相同。

维吉尼亚密码的加密过程

维吉尼亚密码的加密过程可能需要使用一个字母表，这个字母表被称为 Vigenère 方阵。该方阵包含 26 个不同的行，每一行对应一种可能的凯撒密码。方阵中的每一行都相对于前一行循环左移一位。在加密过程中的不同阶段，加密者会从方阵的一个行中选定一个字母表用于加密。选定哪一行则取决于一个重复的关键词。
例如，假设需要加密的明文是 "attackatdawn"，发送加密消息的人选择关键词 "LEMON"，并重复该关键词，使其长度与原文匹配。每个字母表的开头是一串关键字母，行中的其余部分是字母 A 到 Z 的一个偏移序列。此处，加密者只需要 5 个不同的字母表，对应关键词 "LEMON" 中的 5 个字母。

维吉尼亚密码的加密过程

对于明文中的每一个字母，都会取关键词中的对应字母，并使用与该字母对应的行进行加密。在消息的新字符被选中时，关键词的下一个字母会被选取，并沿着对应的行查找匹配消息字符的列头。在对应的 [key-row, msg-col] 的交点处的字母就是加密后的字母。

例如，在加密 "attackatdawn" 中的第一个字母 a 时，取关键词 "LEMON" 中的第一个字母 L，并使用 Vigenère 方阵中的 A 行、L 列相匹配的字母，即 L，作为加密字符。同理，加密明文第二个字母的时候，取关键词 "LEMON" 中的第二个字母 E，会得到字母 X。使用同样的方法，明文 "attackatdawn" 对应的密文为 "LXFOPVEFRNHR"。

Vigenère 方阵

	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z
A	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z
B	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A
C	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B
D	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C
E	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C	D
F	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C	D	E
G	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C	D	E	F
H	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C	D	E	F	G
I	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C	D	E	F	G	H
J	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C	D	E	F	G	H	I
K	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C	D	E	F	G	H	I	J
L	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C	D	E	F	G	H	I	J	K
M	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C	D	E	F	G	H	I	J	K	L
N	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C	D	E	F	G	H	I	J	K	L	M
O	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C	D	E	F	G	H	I	J	K	L	M	N
P	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O
Q	Q	R	S	T	U	V	W	X	Y	Z	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P
R	R	S	T	U	V	W	X	Y	Z	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q
S	S	T	U	V	W	X	Y	Z	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R
T	T	U	V	W	X	Y	Z	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S
U	U	V	W	X	Y	Z	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T
V	V	W	X	Y	Z	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U
W	W	X	Y	Z	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V
X	X	Y	Z	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W
Y	Y	Z	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X
Z	Z	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y

图: Vigenère 方阵的例子

维吉尼亚密码的解密过程

解密过程涉及到查找密文字母在关键词对应行中的位置，然后用该列的标签作为解密字母。例如，在解密密文 "LXFOPVEFRNHR" 中的第一个字母 L 时，查找关键词 "LEMON" 中的第一个字母对应的行，你会在 A 列找到密文的字母 L，因此明文的第一个字母为 a。接着，在关键词 "LEMON" 中的第二个字母 E 对应的行中，找到密文的第二个字母 X 位置的列标签，因此明文的第二个字母为 t。

Vigenère 密码和频率分析

Vigenère 密码的思想，就像所有其他多字母密码一样，是为了掩盖明文中的字母频率，以阻止简单的频率分析。例如，如果 P 在明文为英文的密文中最为频繁，我们可能会猜测 P 对应于英文中最常用的字母 e。但是，通过使用 Vigenère 密码，e 可以在消息的不同点被编码为不同的密文字母，这就使得简单的频率分析失败。

Vigenère 密码的主要弱点

Vigenère 密码的主要弱点在于其密钥的重复性。如果密码分析师正确地猜出了密钥的长度 n，那么密文就可以被视为 n 个交错的凯撒密码，这些密码可以被单独轻易破解。密钥长度可能通过暴力搜索所有可能的 n 值来找出，或者通过 Kasiski 检测和 Friedman 测试来确定密钥长度。

Kasiski 检验

在 1863 年，Friedrich Kasiski 首次公开了对 Vigenère 密码最初的成功攻击。他的方法终结了对明文知识或使用可识别词作为密钥的依赖。然而，Kasiski 并非发现此攻击的第一人。在 1854 年，Charles Babbage 被 John Hall Brock Thwaites 怂恿，解开了 Vigenère 密码。Thwaites 向 Babbage 挑战了一个任务，Babbage 能否通过加密的版本来找到用于加密原文的关键词。

Kasiski 检验

Kasiski 检测法利用了一个事实，那就是有时候由于偶然的情况，重复的单词会使用同样的密钥字母进行加密，从而在密文中形成重复的组。例如，使用关键字 ABCD 进行以下加密：
Key: ABCDABCDABCDABCDABCDABCDABCD Plaintext: cryptoisshortforcryptography Ciphertext: CSASTPKVSIQUTGQUCSASTPIUAQJB
在密文中很容易注意到的重复性，所以 Kasiski 测试将会很有效。CSASTP 的重复之间的距离是 16。如果假设重复的片段代表相同的明文片段，那就意味着密钥长度是 16，8，4，2，或者 1 个字符。

Kasiski 检验

更长的消息使测试更准确，因为它们通常包含更多重复的密文片段。假定密文有两个重复的片段：
Ciphertext: VHVSSPQUCEMRVBVBBBVHVSURQGIBDUGRNICJQUCERVUAXSSR
VHVS 的重复之间的距离是 18。假设重复的片段表示相同的明文片段，则密钥的长度可能为 18，9，6，3，2 或 1 个字符。QUCE 的重复距离是 30 个字符，那意味着密钥长度可能为 30，15，10，6，5，3，2 或 1 个字符。通过取这些集合的交集，我们可以得出最可能的密钥长度为 6，因为 3，2 和 1 的长度短达不现实。

哈希函数简介

哈希函数是可以将任意大小的数据映射到固定大小值的函数，尽管也存在一些支持输出可变长度的哈希函数。哈希函数返回的值被称为哈希值、哈希码、摘要，或者简单的称为哈希。这些值通常用于索引一个固定大小的表，这个表被称为哈希表。使用哈希函数索引一个哈希表被称为哈希或者散列存储寻址。

哈希函数简介

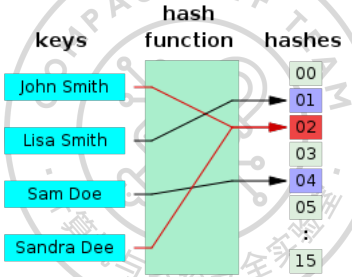
hash
keys
function
hashes
John Smith
Lisa Smith
Sam Doe
Sandra Dee
00
01
02
03
04
05
:
15

图: 哈希表

哈希函数的应用

哈希函数及其相关的哈希表被用于数据存储和检索应用，以便在几乎恒定的时间内访问数据。它们仅需要略大于数据或记录本身所需总空间的存储空间。相比于有序和无序列表的非常数访问时间，以及直接访问大型或可变长度键的状态空间所需的常常是指数级的存储要求，哈希是一种在计算和存储空间效率上都优秀的数据访问方式。

哈希函数的特性

哈希函数的使用依赖于关键字和函数交互的统计特性：最坏情况的行为是难以忍受的，但罕见，而平均情况的行为可以接近最优（即冲突最小）。哈希函数与校验和、校验位、指纹、有损压缩、随机化函数、纠错码和密码在某种程度上相关，且常常被人们混淆。尽管这些概念在某种程度上重叠，但每个概念都有自己的用途和要求，且设计和优化的方式也不同。哈希函数主要在数据完整性方面与这些概念有所不同。

哈希函数的基本概念

哈希函数将一个键作为输入，这个键与一个数据或记录相关联，并被用来在数据存储和检索应用中识别它。键可能是固定长度的，如整数，或是可变长度的，如姓名。在某些情况下，键本身就是数据。输出是一个哈希码，用来索引保存数据或记录的哈希表，或是指向它们的指针。

哈希函数的三个功能

1. 将可变长度的键转换为固定长度（通常是机器字长或更短）的值。通过使用像 ADD 或 XOR 这样的保留奇偶性的运算符按字或其他单元对它们进行折叠。
2. 对键的位进行打乱，使得结果值在键空间上均匀分布。
3. 将键值映射到小于或等于表的大小的值。

哈希函数的两个基本属性

一个好的哈希函数应满足两个基本属性：1) 应该能非常快地进行计算；2) 应该最小化输出值的重复（冲突）。哈希函数依赖于生成有利的概率分布来提高其有效性，将访问时间减小到近乎常数。高表加载因子、病态键集和设计不良的哈希函数可能导致访问时间接近线性地增长与表中项目数量。哈希函数可以设计成具有最佳的最坏情况性能，高表加载因子下的良好性能，以及在特殊情况下，将键完美（无碰撞）地映射到哈希码。实现基于保留奇偶性的位操作（XOR 和 ADD）、乘法或除法。哈希函数必需的附属工具是一种碰撞解决方法，它使用了诸如链表这样的辅助数据结构，或者系统地对表进行探测以找到一个空槽。

哈希表

哈希函数结合哈希表，用于存储和检索数据项或数据记录。哈希函数将跟每个数据源或记录相关联的键转化为哈希码，用于索引哈希表。当需要将项添加到表时，哈希码可能会定位到一个空槽位（也称为存储桶），此时可以将项添加到表的槽位。如果哈希码定位到一个已满的槽位，则需要采取冲突解决策略：新项可能被省略（不添加到表），或者替换旧项，或者通过特定的过程添加到表的其他位置。该过程依赖于哈希表的结构：在链式哈希中，每个槽位是链表或链的头部，碰撞在槽位的项会被添加到链中。链可以随机排列并进行线性搜索，也可以串行排列，或者按频率自我排序以加速访问。在开放地址哈希中，从占用槽位开始按照指定的方式探测表，通常通过线性探测、二次探测或双哈希，直到找到一个空的槽位或者探测完整个表（溢出）。搜索项跟随同一流程，直到找到项、找到一个空槽位或者搜索完整个表（表中没有项）。

哈希函数的专门应用

哈希函数也用于构建大数据集的缓存，这些数据集存储在慢速介质中。缓存通常比哈希搜索表简单，因为任何冲突都可以通过丢弃或写回两个冲突项中的较旧项来解决。哈希函数是 Bloom 过滤器的必要成分，Bloom 过滤器是一种空间效率的概率性数据结构，用于测试一个元素是否属于一集合。哈希的一个特殊情况被称为几何哈希或网格方法。在这些应用中，所有输入的集合是某种度量空间，哈希函数可以被解释为将该空间划分为网格单元的过程。表通常是一个具有两个或更多索引的数组（被称为网格文件、网格索引、存储桶网格等），哈希函数返回索引元组。此原理在计算机图形学、计算几何学和许多其他学科中被广泛应用，用于解决平面或三维空间中许多接近问题，例如在点集中找到最接近对，在形状列表中找到相似形状，在图像库中找到类似图像等。哈希表也被用于实现关联数组和动态集合。

哈希函数的均匀性

好的哈希函数应该尽可能均匀地将预期输入映射到其输出范围。也就是说，输出范围中的每个哈希值都应该以大致相同的概率产生。这个需求的理由是，随着冲突数量的增加——也就是映射到相同哈希值的输入对——基于哈希的方法的成本将急剧上升。如果某些哈希值比其他值更有可能发生，那么更大比例的查找操作将不得不搜索更大的冲突表项集合。

请注意，这个标准只要求值均匀分布，不必以任何意义上的随机。好的随机函数通常（除了计算效率问题）是哈希函数的好选择，但反之不必然成立。

哈希表通常只包含有效输入的一小部分。例如，一个社团的会员名单可能只包含一百个左右的会员名字，出自所有可能名字的非常大的集合。在这些情况下，均匀性标准应该适用于表中可能找到的几乎所有典型的条目子集，而不仅仅是所有可能条目的全局集合。

哈希函数的均匀性

换句话说，如果典型的 m 记录被哈希到 n 个表槽，那么一个哈希空间接受的记录远远超过 m/n 个的概率应该是微乎其微的。特别是如果 m 小于 n，那么很少有哈希空间应该有超过一两条记录。即使 n 比 m 大得多，也几乎不可避免的会有少量的冲突——参见生日问题。

在特殊情况下，如果键是预先知道的并且键集是静态的，那么可以找到一个实现绝对（或者说无冲突的）均匀性的哈希函数。这样的哈希函数被称为完美的。没有算法的方式可以构造出这样的函数——寻找一个如此的函数是一个阶乘函数，它取决于要映射的键的数量和它们被映射到的表槽的数量。在比一小部分键的集合上找到完美哈希函数通常是在计算上不可行的；得到的函数可能比标准哈希函数在计算上复杂，并且对比一个统计性质好，并且可以生成最小碰撞数量的函数，它仅提供了边际效益。

测试和测量

当测试哈希函数时，可以通过卡方检验评估哈希值分布的均匀性。这种检验是一种拟合程度的度量：它将项目在桶中的实际分布与项目的预期（或均匀）分布进行比较。
置信区间（0.95 - 1.05）内的比率表明评估的哈希函数具有预期的均匀分布。
哈希函数可能具有一些技术属性，使得应用时更可能具有均匀分布。其中一个是严格的雪崩准则：每当一个输入比特被取反时，每个输出比特的变化概率都是 50%。这个属性的原因在于，键值空间的选择子集可能具有低的可变性。为了使输出均匀分布，即使一点低变异性也应该转化为输出中的高变异性（即表空间的分布）。每个比特应该有 50%的变化概率，因为如果一些比特不愿改变，那么键就会倾向于那些值。正如文献中所描述的，对于这个特性已经有了标准的测试。

效率

在数据存储和检索应用中，哈希函数的使用是搜索时间与数据存储空间之间的权衡。如果搜索时间无界，一个非常紧凑的无序线性表将是最好的媒介；如果存储空间无界，一个由键值索引的随机可访问的结构会非常大，非常稀疏，但非常快。哈希函数用有限的时间把一个可能很大的键空间映射到一个可以在有限时间内被搜索的可管理的存储空间，不论键的数量。在大多数应用中，哈希函数应该尽量用最小的延迟来计算，并次要地尽量用较少的指令数进行计算。

效率

根据所需的指令数和单个指令的延迟性，计算复杂性会有所不同，其中最简单的是位处理方法（折叠法），其次是乘法方法，最复杂（最慢）的是基于除法的方法。因为冲突应该少见，并且只会造成微小的延迟但其他没有害处，所以通常最好选择较快的哈希函数，而不是需要更多计算但可以保存一些冲撞的哈希函数。

效率

基于除法的实现可能引起特殊关注，因为除法在几乎所有的芯片架构上都是微程序化的。除以一个常数可以被逆转为乘以这个常数的字大小乘法逆数。这可以由程序员完成，也可以由编译器完成。

如果键在被哈希时重复出现，且哈希函数成本较高，可以通过预计算哈希码并将它们与键一同存储来节省计算时间。匹配的哈希码几乎肯定意味着键是相同的。此技术用于游戏程序的置换表，该表存储了一个 64 位的经过哈希的棋盘位置表示。

附加注释

我们可以允许表的大小 n 不是 2 的幂，同时还可以避免执行任何余数或除法操作，因为这些计算有时可能较为昂贵。例如，让 n 远小于 2b。然后考虑一个均匀分布在区间 [0，2b - 1] 的伪随机数生成器函数 P(key)。在区间 [0，n - 1] 上均匀分布的哈希函数是 nP(key)/2b。我们可以通过可能更快的右移位操作替换除法：nP(key) » b。

普遍性

通用哈希（Universal hashing）是一种选择哈希函数 h 的随机化算法，该函数为一系列函数中的一个，而且被选择的方式使得任何两个不同键的碰撞概率为 1/m，其中 m 是期望的不同哈希值的数量——这是与这两个键无关的。通用哈希（在概率意义上）确保了哈希函数的应用就好像它在使用一个随机函数一样，无论输入数据的分布如何。然而，它会有更多的碰撞比完美哈希，并可能需要比特殊目的哈希函数更多的操作。

适用性

哈希函数可以应用于多种情况。一个只允许特定表大小，只支持到一定长度的字符串，或者不能接受种子（即不允许双哈希）的哈希函数没有那些可以做到这些的哈希函数有用。

确定性

一个哈希过程必须是确定的——意味着对于给定的输入值，它必须总是生成相同的哈希值。换句话说，它必须是待哈希数据的函数，具有严格的数学意义。这项要求排除了那些依赖于外部变量参数的哈希函数，例如伪随机数生成器或者时间。这也排除了一些依赖于被哈希对象内存地址的函数，因为在执行过程中地址可能会发生变化（如在使用了某种垃圾收集方法的系统上可能发生），虽然有时候重新哈希对象是可能的。

公钥密码系统

RSA 是一种公钥密码系统，也是最古老的密码系统之一，在安全数据传输中得到了广泛的应用。"RSA" 这个缩写来自于 Ron Rivest, Adi Shamir 和 Leonard Adleman 的姓氏，他们在 1977 年公开描述了这个算法。1973 年，英国政府通信总部（GCHQ）的英国数学家 Clifford Cocks 即英国信号情报机构，秘密开发了一个等同的系统。这个系统在 1997 年解密。[7]

公钥密码系统的工作原理

在公钥密码系统中，加密密钥是公开的，并且与解密密钥（保密）不同。RSA 用户根据两个大质数和一个辅助值创建并发布公钥。质数是保密的。任何人都可以通过公钥加密消息，但只有知道质数的人才能解码。

RSA 的安全性和效率

RSA 的安全性依赖于分解两个大质数的乘积的实际难度，也就是" 分解问题"。破解 RSA 加密被称为 RSA 问题。"RSA 问题" 是否与" 分解问题" 一样困难，这个问题尚未解决。目前没有公开的方法能在使用足够大的密钥的情况下打败这个系统。

RSA 是一个相对较慢的算法，因此通常不会被用来直接加密用户数据。更常见的是，RSA 被用来传输对称密钥密码学的共享密钥，然后用于大量的加密-解密。

Diffie 和 Hellman 的贡献

1976 年，Whitfield Diffie 和 Martin Hellman 首次发布了非对称公钥-私钥加密系统的概念。他们还引入了数字签名和试图应用数论。他们的公式使用了一个从某个数的指数运算，模一个素数而来的共享密钥。然而，他们并未解决实现单向函数的问题，可能是因为当时因子分解的难度并未得到深入研究。

RSA 算法的诞生

在一年的时间里，麻省理工学院的 Ron Rivest、Adi Shamir、Leonard Adleman 多次尝试创建一种难以逆向求解的函数。作为计算机科学家的 Rivest 和 Shamir 提出了许多潜在的函数，而身为数学家的 Adleman 负责找出他们的弱点。他们尝试了多种方法，包括" 背包算法" 和" 排列多项式"。在 1977 年 4 月，他们在一个学生的家中度过了逾越节，在午夜左右返回他们的家中。Rivest 躺在沙发上，手中拿着一本数学教材，开始思考他们的单向函数。他花了整晚的时间构思自己的想法，到天亮时他已经完成了大部分的论文。现在这个算法被称为 RSA 算法——他们的姓氏首字母的联合体。

Clifford Cocks 的贡献和 Kid-RSA

英国情报机构政府通讯总部的数学家 Clifford Cocks 在 1973 年的一份内部文件中描述了一个等价的系统。然而，考虑到当时需要进行实现的计算机相对昂贵，人们主要认为这是个好奇心产物，就已知的公开信息来看，它从未被使用过。他的发现直至 1997 年才被公开，因为它被归为最高机密。

Kid-RSA（KRSA）是一个简化的，不安全的公钥密码系统，于 1997 年发布，设计目标是教学用途。一些人认为，学习 Kid-RSA 可以对 RSA 及其它公钥密码系统有深入的理解，类似于简化的 DES。

RSA 算法

RSA 算法包括四个步骤：秘钥生成、秘钥分发、加密和解密。
RSA 的基本原理在于，我们可以找到三个非常大的正整数 e，d 和 n，使得对所有的整数 m（0 m < n）：

$$(m^e)^d \equiv m \mod n$$

RSA 算法

即使知道 e 和 n，甚至 m，找出 d 也可能极其困难。这里的三竖线 ( ) 表示模同余。
也就是说，当你将 $(me)d$ 和 m 都除以 n 后，它们的余数是一样的。
另外，一些操作中，改变两次幂运算的顺序可能更方便，而且这种关系也意味着

$$(m^d)^e \equiv m \mod n$$

RSA 涉及一个公钥和一个私钥。公钥可以被所有人获知，用于加密消息。目标是用公钥加密的消息只能通过私钥在合理的时间内解密。公钥由整数 n 和 e 表示，私钥由整数 d 表示（尽管在解密过程中也使用了 n，所以它也可能被认为是私钥的一部分）。m 代表消息（之前用下文解释的某种技术进行了预处理）。

RSA 算法的密钥生成

- 选择两个大的素数 p 和 q，以随机方式选择，且 p、q 差异较大以增加因数分解难度。保密 p 和 q。
- 计算 $n = pq$。n 是公钥和私钥的模，其长度通常以比特表示为密钥长度。n 作为公钥的一部分被公开。
- 计算 $\varphi(n)$，φ 是卡迈克尔总函数。由于 $n=pq$，$\varphi(n) = \mathrm{lcm}(\varphi(p), \varphi(q))$。且由于 p 和 q 是素数，所以 $\varphi(p) = \varphi(p) = p - 1$，同样 $\varphi(q) = q - 1$。因此 $\varphi(n) = \mathrm{lcm}(p - 1, q - 1)$。

RSA 算法的密钥生成

- lcm 可以通过欧几里得算法计算，因为 lcm(a, b) = |a*b| / gcd(a, b)。 (n) 被保密。
- 选择一个整数 e，使得 2 < e < (n) 并且 gcd(e, (n)) = 1；即，e 和 (n) 是互质的。e 与短比特长度和小海明权重有更有效的加密结果，最常选择的 e 值为 65537。e 最小可能值为 3，但在一些设置中，e 选择小是不安全的。
- e 是公钥的一部分公开的。求解 d，d e—1 (mod (n))；即，d 是模 (n) 下 e 的模逆。
- 这意味着，求解方程 de 1 (mod (n)) 的 d；d 可以通过扩展的欧几里得算法有效的被计算，得到 d 是 Bézout's 恒等式的一个系数形式，由于 e 和 (n) 是互质的。
- d 被保密作为私钥的指数。公钥由模 n 和公开的指数 e 组成。私钥由私人的指数 d 组成，d 必须被保密。p，q 和 (n) 也需要保密因为可以用他们来计算 d。

- 在原始 RSA 论文 [1] 中，选择用欧拉函数 (n) = (p − 1)(q − 1) 代替 (n) 来计算私人指数 d。由于 (n) 总是被 (n) 整除，所以算法同样适用。
- 对于模 pq 的整数乘法群应用拉格朗日定理，任何满足 d e 1 (mod (n)) 也满足 d e 1 (mod (n)) 的 d。但是，模 约会话胥 d 有时候可能会比需要的结果大（即，d > (n)）。大部分 RSA 实现接受使用任意两种方法生成的指数（如果他们使用私人指数 d，相比使用中国剩余定理优化解密方法）。但是，有些标准如 FIPS 186-4（第 B.3.1 节）可能需要 d < (n)。任何超过大小的私人指数无法满足这个准则可能总是通过约化模 (n) 获得一个更小的等价指数。

- 由于 $(p-1)$ 和 $(q-1)$ 的公因子存在于 $n-1=pq-1=(p-1)(q-1)+(p-1)+(q-1)$ 的分解中，建议 $(p-1)$ 和 $(q-1)$ 只有非常小的公因数，如果有的话。
- 注意：原始 RSA 论文的作者通过选择 d 然后计算 e 作为模 (n) 下 d 的模逆来进行密钥的生成，而大部分当前的 RSA 实现例如遵循 PKCS#1 的做法是反过来（选择 e 然后计算 d）。由于选择的密钥可以较小，而计算的密钥通常不是，RSA 论文的算法优化了解密相比加密，而现代算法优化了加密。

密钥分发

假设 Bob 希望向 Alice 发送信息。如果他们决定使用 RSA，Bob 必须知道 Alice 的公钥来加密信息，而 Alice 则需要使用她的私钥来解密信息。
为了让 Bob 能够发送他的加密信息，Alice 通过一个可靠的，但不一定是秘密的途径，向 Bob 发送她的公钥 (n, e)。Alice 的私钥 (d) 永不分发。

加密

在 Bob 获取 Alice 的公钥后，他可以向 Alice 发送一条消息 M。
为了做到这一点，他首先将 M（严格来说，是未填充的明文）变成一个整数 m（严格来说，是填充后的明文），使得 $0 \leq m < n$，他使用一个大家都认可的可逆协议，也被称为填充方案。然后他利用 Alice 的公用密钥 e 计算出密文 c，满足

$$c \equiv m^e \pmod{n}.$$

这个过程使用模幂运算，即使对非常大的数，也能相当快的完成。Bob 然后发送密文 c 给 Alice。注意，至少有九个 m 的值能得到一个等于 m 的密文 c，但在实际中这种情况很少发生。

解密

Alice 可以使用她的私钥指数 d 从密文 c 中恢复出 m。

$$c^d \equiv (m^e)^d \equiv m \pmod{n}.$$

掌握了 m，她就可以通过逆转填充方案恢复出原始的消息 M。

RSA 加密和解密的范例

这里是一个 RSA 加密和解密的例子。这里使用的参数是人工设定得很小，但是也可以用 OpenSSL 来生成和检测一个真实的密钥对。
选择两个不同的素数，如

$$p = 61, q = 53.$$

计算

$$n = p * q,$$

得到

$$n = 61 * 53 = 3233.$$

RSA 加密和解密的范例

计算 Carmichael's totient 函数，得到

$$\lambda(3233) = lcm(60, 52) = 780.$$

选择任意一个 $1 < e < 780$，且与 780 是互质数的数。让

$$e = 17.$$

计算 d，它是 e (mod (n)) 的乘法逆元，因此

$$1 = (17 * 413) mod 780.$$

密钥和加密函数

公钥是 (n = 3233, e = 17) 对于一个填充了的明文消息 m，它的加密函数是

$$c(m) = m^e mod n = m^{17} mod 3233.$$

私钥是 (n = 3233, d = 413) 对于一个已加密的密文 c，它的解密函数是

$$m(c) = c^d mod n = c^{413} mod 3233.$$

密钥和加密函数

例如，如果想要加密 m = 65, 可以计算

$$c = 65^{17} mod 3233 = 2790.$$

解密 c = 2790, 可以计算

$$m = 2790^{413} mod 3233 = 65.$$

解密和优化

私钥的组成部分 dp, dq 和 qinv 的计算方法如下:

$$dp = dmod(p-1) = 413mod(61-1) = 53,$$

$$dq = dmod(q-1) = 413mod(53-1) = 49,$$

$$q_{inv} = q^{-1}modp = 53^{-1}mod61 = 38,$$

然后有

$$(q_{inv} * q)modp = 38 * 53mod61 = 1.$$

解密和优化

下面说明了如何使用 dp, dq 和 qinv 来高效地进行解密（通过选择合适的 d 和 e 对，加密就可以高效地进行）：

$$m_1 = c^{d_p} mod p = 2790^{53} mod 61 = 4,$$

$$m_2 = c^{d_q} mod q = 2790^{49} mod 53 = 12,$$

$$h = (q_{inv} * (m_1 - m_2)) mod p = (38 * (-8)) mod 61 = 1,$$

$$m = m_2 + h * q = 12 + 1 * 53 = 65.$$

消息签名

假设 Alice 使用 Bob 的公钥向他发送了一条加密的消息。在消息中，她可以自称为 Alice，但 Bob 无法确认消息的真正来源是 Alice，因为任何人都可以使用 Bob 的公钥向他发送加密的消息。为了验证消息的来源，RSA 也可以用来对一条消息进行签名。假设 Alice 想要向 Bob 发送一条签过名的消息。她可以使用自己的私钥来做到这一点。她产生消息的哈希值，将其提升为 d 次幂（模 n）（就像她解密消息时做的那样），并将其作为" 签名" 附加在消息上。

消息签名

当 Bob 收到签过名的消息时，他使用相同的哈希算法和 Alice 的公钥。他将签名提升为 e 次幂（模 n）（就像他加密消息时做的那样），并将所得的哈希值与消息的哈希值进行比较。如果一致，他就知道消息的来源持有 Alice 的私钥，并且消息自发出以来没有被篡改。

这是乘方的规则：

$$h = \mathrm{hash}(m),$$

$$(h^e)^d = h^{ed} = h^{de} = (h^d)^e \equiv h \pmod{n}.$$

消息签名

因此，公私钥可以互换，即密钥对的私钥可以按以下方式使用：
解密：只有持有者可以解密，任何人都可以使用公钥进行加密（非对称加密传输）。
加密：任何人都可以解密，但只有一个人可以加密；这提供了数字签名。

参考文献

[1] Wikipedia, Wikimedia Foundation. Substitution Cipher[EB/OL]. 2023. https://en.wikipedia.org/wiki/Substitution_cipher.

[2] CRAWFORD D, ESTERL M. At Siemens, witnesses cite pattern of bribery[J]. The Wall Street Journal, 2007.

[3] BRUEN A A, FORCINITO M A. Cryptography, Information Theory, and Error-Correction: A Handbook for the 21st Century[M]. John Wiley & Sons, 2011.

[4] MARTIN K M. Everyday Cryptography[M]. Oxford University Press, 2012.

[5] SMITH L D. Cryptography: The Science of Secret Writing[M]. Courier Corporation, 1955.

[6] RODRIGUEZ-CLARK D. Vigenère Cipher[EB]. Crypto Corner. 2017.

[7] Wikipedia. RSA (Cryptosystem)[EB/OL]. Wikimedia Foundation. 2023. https://en.wikipedia.org/wiki/RSA_(cryptosystem).